Mobility Analytics

Team 5: Born Strikers

Aakriti Istwal, 1871754
Manmohit Singh Slaich, 1836523
Pavani Rajula, 1870208
Progha Labani Das, 1871821
Serxhina Kutrolli, 1871912
Sharan Shyamsundar, 1870221

University of Mannheim, Germany

Abstract. For a decade, 'Surge pricing' or 'Dynamic pricing' has been widely utilized by airlines and hotels, which is considered a systematic method of price management. More recently, there has been a boom in the ride/cab-hailing service platforms where higher attention is being paid to price decisions due to demand fluctuation which reflects supply and demand imbalances in the transportation industry. Ultimately, to make the transportation networks more reliable and efficient, the various businesses introduced cab aggregator service to capture surge pricing that can be used more accurately to assign vehicles, saves riders money and time, and gives profit-making awareness to drivers. Due to the high demand for mobility solutions as well as cab aggregators, there has been massive development in data collected from commercial vehicles with major ride-sourcing companies.

Our team of the Data Mining-1 Project concentrates on the attempts to create a "Mobility Analytics" solution which aims to build a predictive model that helps in predicting surge price type proactively by using various attributes. For this project, our team used raw data from one of the competition platforms and various approaches are being used to understand data through the dataset, data exploration, and target variable analysis. Then we analyzed various methodologies such as data pre-processing and modeling to plan, manage and execute our project. Finally, these models are being evaluated to check the accuracy of predicted results followed by a conclusion.

Keywords: Machine Learning · Classification · Predictor Variable

1 Introduction

Transportation industry is thriving with ride share companies who are spiking costs at seemingly every turn and the interplay of supply and demand leads to take the real time conditions into account before suggesting a price to the customer. Factors such as rush hour, weather changes or event blockers can increase the demand of rides, on the other hand, supply of riders may plummet, and this is when cab prices are surged.

Here, we will explore one such cab aggregator service - "Sigma cab private limited", providing facilities in assistance with different service providers for almost a year. Using different data mining methods we will analyse the gathered data of surge pricing type allocated to the customers and build a model which can help in pro-actively predicting surge pricing type using the attributes such as trip distance , cab type, customer lifestyle details and much more. In order to efficiently allocate cabs, save customers time and provide profitable insights to riders; better predicted surge price definitely proves to be helpful .

2 Data Understanding

2.1 Dataset

For this problem, we are using a dataset that is available on the contest page [1]. A dataset of around 0.2 million trips is considered in this case. The customers search for cabs available from the cab provider and then reserve one for a trip. While seeking for the cab, the service provider considers parameters such as journey distance, client rating, recent cancellations (if any) etc. Based on these attributes, the company then determines a particular surge pricing type for each trip. Each trip booked by a customer is recorded with a unique Trip ID, there are no instances where a customer has looked for a cab but not booked it. This activity has been in operation for some months now and the company has found that to make a profit and give customers the right price they have to give each trip a particular level of surge pricing.

The metadata consists of 12 features that have their respective representations as shown in the table :

1. Trip_Distance - The distance for the trip requested by the customer.
2. Type_of_Cab - Category of the cab requested by the customer.
3. Customer_Since_Months - Customer using cab services since n months; 0 month means current month.
4. Life_Style_Index - A proprietary index created by Sigma Cabs showing the lifestyle of the customer based on their behavior.
5. Confidence_Life_Style_Index - Category showing confidence on the index mentioned above.
6. Destination_Type - Sigma Cabs divides any destination into one of the 14 categories.
7. Customer_Rating -Average of lifetime ratings of the customer till date.
8. Cancellation_Last_1Month - Number of trips canceled by the customer in the last 1 month. 9

9. Var1, Var2, and Var3 - Continuous variables masked by the company. Can be used for modeling purposes.
10. Gender - Gender of the customer.

Surge_Pricing_Type is the predictor variable which is of 3 types (1, 2 & 3) and is derived based on trip features mentioned above.

2.2 Data Exploration

To understand the data, we looked at each variable and tried to understand their meaning and relevance to this problem and how we can use them to find the predictor variable using multiple exploratory data analysis techniques. We found out that there are features with up to 0-15% missing values, which need to be handled to ensure that the missing data process is not biased and hides an inconvenient truth and there is the Var1 feature which has about 51% missing values but has a reasonable correlation with the target. There are also a few visible outliers that need to be handled. Figure 1 shows the correlation heatmap which shows how significant the correlation is between variables.

It can be inferred from the heat map that all the features have less correlation with the Surge_Pricing_Type. Among which Type_of_Cab has the highest followed by Cancellation_Last_1Month, Customer_Rating, and Trip_Distance and Destination_Type and Gender having the least. Trying dropping few features by this analysis or by using machine learning techniques to decide which features are to be considered for the prediction.

2.3 Target Variable Analysis

The target variable, Surge_Pricing_Type is a categorical variable that indicates the surge price class for a specific trip. The Surge_pricing_type consists of 3 types, the company has not mentioned which levels mean because of security reasons. Surge Pricing in general means when the demand for cabs is high, and supply low, how much surge charge should the company add to the base price so that only those who can match the price will book and they can match the demand and supply. But the company might not apply the same surge level for each trip. Using the top features with high correlation, below are the insights derived by plotting various plots, which helped to understand the target variable better.

- Customers who have requested Cab A, are mostly categorized under Type 1 and Cab B, C as Type 2 and D, and E as Type 3. Assuming that Cab A is the basic cab type and E is the premium cab it indicates that Type 1 is the lowest pricing type and Type 3 is expensive.
- Data provided has customers from the past 10 months, out of which it indicates that new customers who have joined and up to 6 months approximately are considered for Type 1 and the oldest for the Type 3 which means that new customers are given cabs at an affordable price which helps to maintain the relation with the customers.

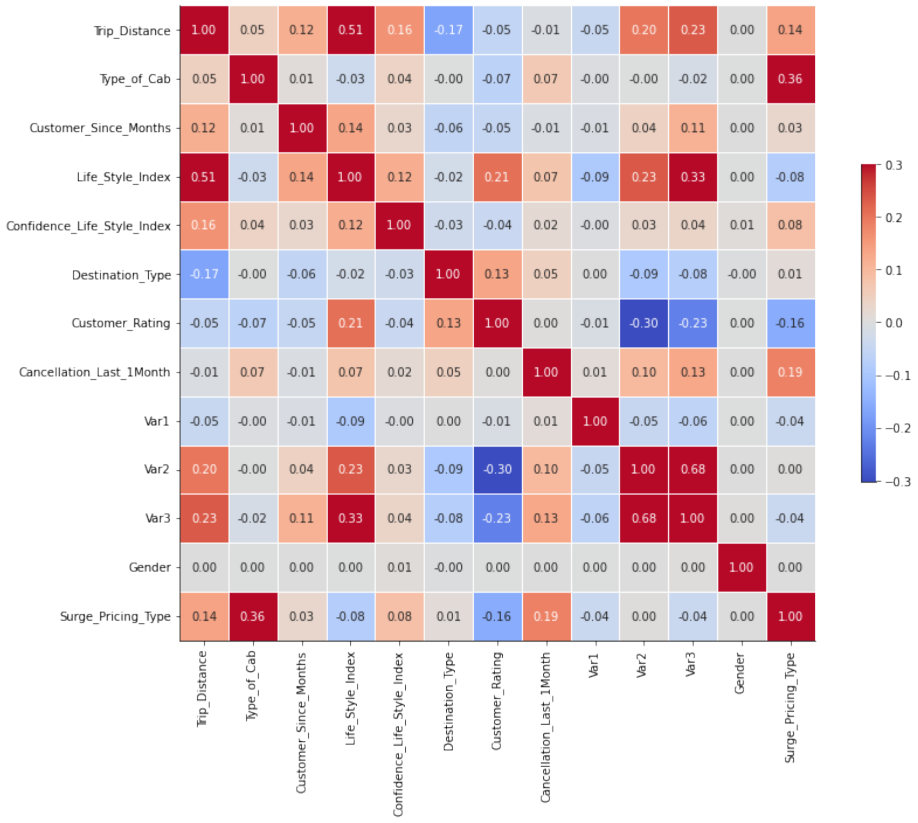

Fig. 1. Feature Correlation - Heatmap

- Long-distance trips are very expensive.
- Customers with high customer ratings and high Life_Style_Index are given affordable prices and are considered under Type 1.
- Customers with more number of cancellations in the last month will have a greater surge price.

In the following sections, we would see how these features are used to predict the Surge_Pricing_Type using multiple machine learning techniques.

3 Methodology

3.1 Data Preprocessing

The raw data is provided by the competition platform. They have already split the dataset into train and test. In general, real-world data is not always complete and error-free. The data set lacked some attribute values of interest, and some

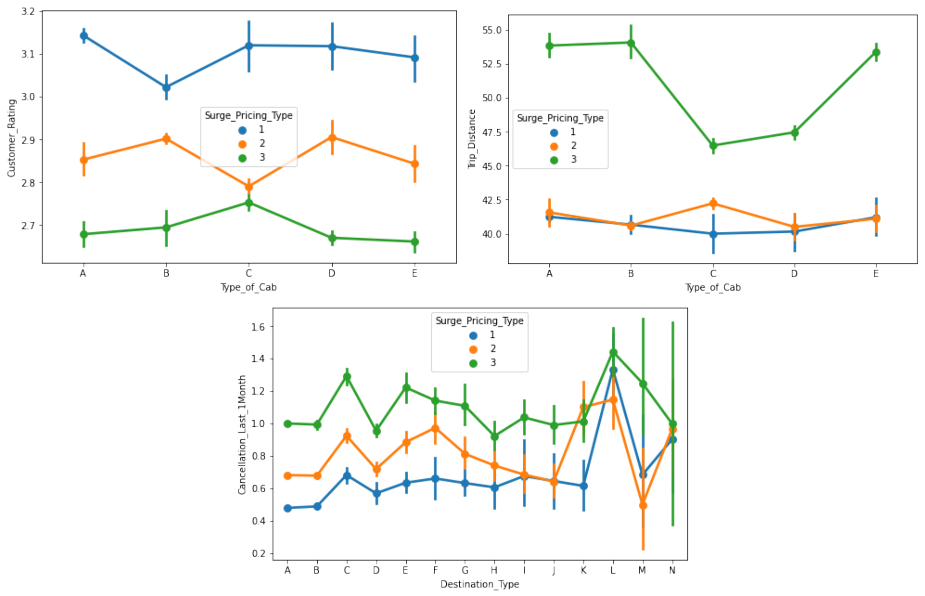

Fig. 2. Plots from the working notebook

attributes were not ready for model preparation. A more appropriate data format generation for the model training involves various treatments.

Missing Value Treatments
It is observed that we have a couple of attributes that have missing values. There are numerous ways in which these can be treated. We went forward and created 3 different scenarios where the missing values are treated in different ways and then trained models on these different scenarios.

- **Preprocess 1:** We replaced the categorical missing values by creating a new category named *Unknown*. And replaced the numeric missing values with *0*.
- **Preprocess 2:** In this case, the *median* of the attributes replaced the missing ones in numeric types. And the *most frequent category* for each categorical attribute replaced the missing values in categorical types.
- **Preprocess 3:** To perform the imputation, we used the KNN Imputer. It works by finding the *K closest neighbors* to the observation with missing value and then imputing them based on the non-missing values in the neighbors.

All 3 preprocessing techniques were applied to both the train and test dataset.

Label Encoding
The categorical type variables were then label-encoded in each of the scenarios,

using the already available function in the sklearn package [2]. This was done because the machine learning models employed do not allow string data types.

Train Test Split
Lastly, the training set is split into a training set and a validation set using an 80:20 split.

Feature Scaling
During the model fitting without tuning, the data is scaled using a StandardScaler. While hyperparameter tuning, other scalars like MinMaxScaler, RobustScaler and MaxAbsScaler along with Standard scalar were put in the grid search to find the best scaler for the models.
StandardScaler normalizes the columns of the data sets so that the mean of the column becomes close to 0 and the standard deviation becomes close to 1. MinMaxScaler transforms features by scaling each feature to a given range. RobustScaler scales features using statistics that are robust to outliers. MaxAbsScaler scales each feature by its maximum absolute value. [3]

3.2 Modelling

The overall purpose of this project was to determine which surge price category a customer will fall into when booking a cab depending on their profile. As a result, this was a classification problem. To accomplish this, we used the dataset's attributes to train a classifier model. In this case, we used the preprocessed data to train traditional Machine Learning models.[5]

To begin, we utilized scikit-learn [4], a standard Python program for data analysis. Without any hyperparameter adjustment, we established pipelines for each of the machine learning models listed below using a standard scaler and examined the results on the validation and test sets. The best three models were chosen after the results are computed, and hyperparameter tuning was performed on them using grid search and 5-fold cross-validation. We also found the best scaler out of the ones mentioned above in every grid search. This method was repeated for each of the three preprocessing scenarios described above.

Classification Models

Baseline - Dummy Classifier is a type of classifier which does not generate any insight about the data and classifies the given data using only simple rules, and in this case it classifies everything into the majority class. This classifier is a model that makes predictions without trying to find patterns in the data and is used as a baseline to compare against more complex classifiers.

Random Forest is an estimated Classifier used for a large number of different decision trees in such ways that fits this trees on various sub-samples of the dataset and use the averages to improve the prediction accuracy and to control over-fitting. The hyperparameters we are tuning in our case are n_estimators, criterion, max_depth, min_samples_split and min_samples_leaf.

K - Nearest neighbors is one of the simplest algorithms of supervised machine learning. It simply calculates the distance of a new data point to all other training data points. Then selects the K-nearest data points and finally it assigns the data point to the class to which the majority of K data points belong. The hyperparameters we are using in this case are n_neighbours, weights, p,

SVM is a set of supervised learning methods used for classification, regression and outliers detection. It is very effective in high dimensional spaces, even in the cases where the number of dimensions is greater than the number of samples. It also uses a subset of training points in the decision function, which is called support vector. Speaking for our case the hyperparameters that we are using are C, gamma and kernel.

Naive Bayes classifier is a probabilistic machine learning model that is used for classification tasks. This algorithm is based on the Bayes theorem and shows the conditional independence between every pair of features and the value of the class variable. For our case we are using the Gaussian Naive Bayes, which assumes the probability of features to follow the Gaussian distribution.

Decision Trees are Flow Charts that help in making a decision based on previous experience. It is a non-parametric supervised learning method used for classification and regression and it´s goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Hyperparameter Optimization

Halving Grid Search CV with repeated Stratified K-fold is a strategy who search over specified parameter values with successive halving and starts evaluating all the candidates with a small amount of resources and iteratively selects the best candidates, using more and more resources, also combined with a repeated Stratified K fold, which means that the k-fold cross-validation process is repeated multiple times and reports the mean performance across all folds and all repeats and can help reducing the bias in the model´s estimated performance. To conclude using The Halving Grid Search with repeated Stratified K-fold [6] creates a better performance for the model and gives the best results. In our case a 5- Fold cross-validation with 2 repeats was used to estimate the model performance, this means that (2 * 5) or 10 different models were fitted and evaluated.

4 Evaluation & Results

4.1 Evaluation Framework

As previously stated, each preprocessing's training dataset was split into train and validation using a fixed train test split of 80:20 and stratification over the target variable. The competition used accuracy as a criterion for judging the results. The predictions had to be uploaded on the website in order to get the test set's scores. However, as can be seen, the target variable is not evenly balanced. As a result, coupled with accuracy, the validation set's statistic was the macro average F1-score. Each attribute's importance score was computed using SelectKbest.

4.2 Results

Based on our 2 step approach for evaluating the results, one without-hyperparameter tuning and one with-hyperparameter tuning, we observed multiple results for the respective models. The scores were compared side by side for the validation set, obtained from a train test split, and the test set provided by the competition itself.

To begin with, F1 scores for the validation set from table 5 were transparent enough to conclude that the best scores were obtained after tuning all the models; with Random Forest leading the board. Furthermore, since the metric used in the problem statement was accuracy, hence, our results were compared based on the accuracy scores of the models. The without hyperparameter tuning phase from table 2 represents a separation between the scores for the validation and the test set and the highest observed score was from random forest approximating to 0.69, while decision trees ranked the lowest for the test set. Table 4 displays the tuning phase by observing the top 3 models: random forest, knn, svm, tuned on the best parameter combination. The side by side comparison of these models based on accuracy scores again proved that Random Forest contributed the best results for the unseen set of records.

Hence, ultimately, our best results for this dataset with respect to the surge price prediction were obtained by using Random Forest with appropriate hyperparameter tuning.

Validation Macro Avg F1-Score	Validation Accuracy	Test Accuracy
0.2007	0.57505	0.42758

Table 1. Baseline Scores

Preprocessing	Random Forest		KNN		SVM	
	Validation	Test	Validation	Test	Validation	Test
Preprocess 1	0.6914	0.6909	0.5932	0.59726	0.6908	0.6902
Preprocess 2	0.6801	0.6822	0.6088	0.61425	0.6817	0.6823
Preprocess 3	0.6916	0.6899	0.6158	0.61618	0.6862	0.6855

Preprocessing	Naïve Bayes		Decision Trees	
	Validation	Test	Validation	Test
Preprocess 1	0.5570	0.5577	0.5751	0.57640
Preprocess 2	0.6351	0.6388	0.5568	0.56038
Preprocess 3	0.6468	0.64861	0.5721	0.57246

Table 2. Accuracy Scores without tuning

Preprocessing	Random Forest	KNN	SVM
Preprocess 1	0.6797	0.5816	0.6772
Preprocess 2	0.6663	0.5974	0.6635
Preprocess 3	0.6775	0.6035	0.6693

Table 3. Macro Average F1 Scores on Validation W/O hyper tune

Preprocessing	Random Forest		KNN		SVM	
	Validation	Test	Validation	Test	Validation	Test
Preprocess 1	0.6996	0.69764	0.6892	0.6884	0.6924	0.6913
Preprocess 2	0.6867	0.68769	0.6785	0.6805	0.6764	0.6777
Preprocess 3	0.6947	0.69389	0.6809	0.6837	0.6858	0.6844

Table 4. Accuracy Scores with tuning

Preprocessing	Random Forest	KNN	SVM
Preprocess 1	0.6873	0.6761	0.6758
Preprocess 2	0.6711	0.6623	0.6575
Preprocess 3	0.6796	0.6657	0.6680

Table 5. Macro Average F1 Scores on Validation With hyper tune

4.3 Error Analysis

The model's errors were then briefly examined using the confusion matrix shown in below Figure 3. When looking at the best-performing model's predictions after tuning, it's clear that the most common error it made was predicting the high and low classes of surge pricing for a category that falls into the majority class medium. This could be due to the lack of equally balanced categories.

5 Conclusion

In today's world, many cab services struggle to quote the correct price to their customers. They must generate a profit while still maintaining their customer base. However, with enough data on the customer's profile and travel information, machine learning approaches can at least partially overcome this problem. Throughout the research, we used a variety of machine learning approaches as well as several preprocessing techniques to get results that were much better than our baseline. Random Forest Classifier produced the best results, particularly in the 1st Preprocessing scenario. With its accuracy score of 0.69764 on the test set it is the best suited model. Other machine learning models outperform the baseline too, although not all of them are as good as Random Forest.

One way to improve in the future is to include gradient boosting models like XG Boost, Ada Boost, and LGBM in the set of machine learning models. These

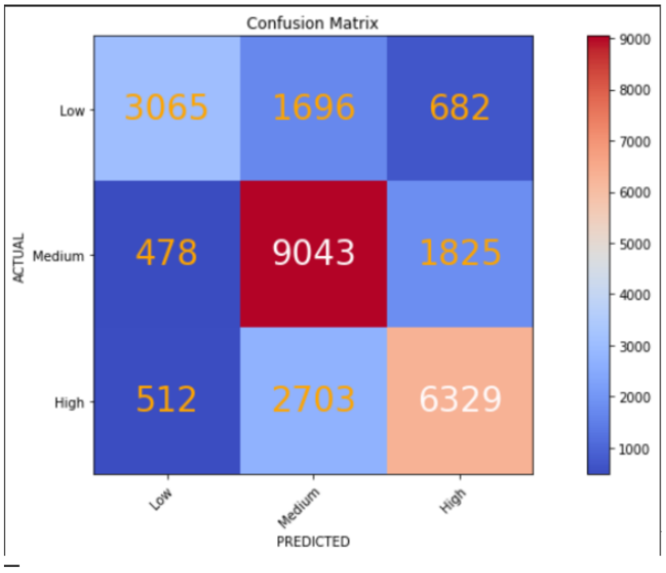

Fig. 3. Tuned Random Forest on Validation Set

types of models Boosting methods use a sequential strategy to integrate numerous weak learners and improve observations iteratively. This method aids in the reduction of high bias in machine learning models. Collecting more information, such as more consumers and features could also help in improving the models. For example, the number of cabs booked in the previous year, the average cab wait time, and cancellations in the previous month can be extended to three months.

To summarize, Machine Learning models assist the organization in accurately classifying clients into the appropriate surge pricing type category. The accuracy of these predictions is adequate. However, more research is needed if companies need to leverage machine learning models for tasks like these.

References

1. Janata Hack : Mobility Analytics Contest Page
   https://datahack.analyticsvidhya.com/contest/janatahack-mobility-analytics/
2. Scikit Learn Label Encoder
   https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
3. Scikit Learn Scalers
   https://scikit-learn.org/stable/modules/preprocessing.html
4. SciKit Learn
   https://scikit-learn.org/stable
5. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011
6. Repeated-k-fold-cross-validation
   https://machinelearningmastery.com/repeated-k-fold-cross-validation-with-python/